# RETAIN: BUILDING A CONCEPT RECOMMENDATION SYSTEM THAT LEVERAGES SPACED REPETITION TO IMPROVE RETENTION IN EDUCATIONAL SETTINGS

BY

SHILPA SUBRAHMANYAM

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor ChengXiang Zhai

# Abstract

There is a glaring lack of focus on long-term retention in today's educational paradigms. Moreover, research in the area of learning, memory, and specifically, promoting long-term retention has produced several robust and experimentally validated principles. A lot of this work can be leveraged to place some much-needed emphasis on long-term retention in educational settings.

One such principle is spaced repetition – a technique that has been empirically proven to promote long-term retention. The applications of current spaced repetition algorithms are limited to *atomic* concepts – concepts that don't have any conceptual dependencies. In order to apply current spaced repetition formulae to more general contexts, we need to develop a system that can take conceptual dependencies into account.

In this paper, we propose a framework called Retain that does exactly this. Retain is a system that can be used in virtually *any* educational context – not just contexts that solely involve atomic concepts (i.e. learning vocabulary terms). It is a concept recommendation system that provides students with suggestions about when to review various concepts based on their understanding of parent concepts and the principle of spaced repetition. The results produced by Retain as well as the rules upon which Retain was built were evaluated by a group of teachers and were overwhelmingly favored over other concept recommendation baselines.

*For my family*
*Amma, Appa, and Varsha*
*Without whom none of this would be possible*

# Acknowledgments

I'd like to thank my adviser, Dr. ChengXiang Zhai, for his guidance and advice throughout this process. I'd also like to thank three of my TIMAN group mates, Assma Boughoula, Fareedah Alsaad, and Chase Geigle, for their support and help as I worked on this thesis. Thank you to Kristen Vaccaro for generously taking the time to help me design the user study for this thesis. Many thanks to Dr. Tandy Warnow and Mike Nute for teaching me about how to approach computer science research. I'd like to thank Dr. Craig Zilles; I learned a lot about teaching and education within the domain of computer science while I worked as his teaching assistant during my last year at UIUC.

Thank you to my mother and sister, Viji and Varsha Subrahmanyam, for their unfailing support and love. I'm also grateful to the friends I've made over the course of my time at UIUC for their support and good humor. Thanks in particular to the friends who served as sounding boards as I worked on this thesis.

Last but certainly not least, I am incredibly grateful to my father, Dr. Ramesh Subrahmanyam, for his constant encouragement, love, and guidance as I navigated through these last five years at UIUC.

# Contents

# Chapter 1

# Introduction

In this paper, we present Retain: a concept review recommendation system. Retain leverages spaced repetition, and by extension, the psychological spacing effect, to help students learn more effectively and ensure improved long-term retention of concepts.

The work detailed in this paper was motivated by a series of broad research questions: (1) How can we improve retention in current educational contexts? (2) How can we leverage existing research in learning and cognitive psychology in the interest of improving student retention? (3) How can we use computer science to solve these problems?

## 1.1 Problems with Retention in Educational Contexts

Halpern et al. identify transfer and long-term retention as the primary objectives of a formal education [1]. Unfortunately, there is a glaring lack of focus on long-term retention (and by extension, transfer) in today's educational paradigms.

A lot of courses are structured in ways that incentivize cramming; they accidentally propagate the notion that comprehension at the time of exams and quizzes is important (no matter how short-lived) and retention is not. This is done through the the standard assessment schemes that test content once or twice throughout the semester with minimal revisitation of content. This notion is also propagated through the traditional structure of course sequences; courses are organized into independent bodies of ideas. Oftentimes, concepts that are presented in one course are never revisited in subsequent courses. This incentivizes students to cram for course midterms and finals. Because there is no explicit incentive to retain this information for future years, students resort to behavior that saves them time and doesn't require them to be proactive – cramming. Cramming is helpful in the short-term, but it is the enemy of long-term retention.

There has been a lot of work that has been done in the field of learning and retention. Unfortunately, to date, even the most robust findings have been ignored in practice. Halpern et al. offer their own theories for why this might be the case: perhaps the teachers feel that their teaching methods are already effective and as such, they do not need to seek guidance from research in the relevant fields. Moreover, we assert that it is somewhat difficult to implement some of the findings found in learning science research. Spaced learning is an intuitive concept for a teacher to understand, but it's difficult and tedious to manually implement. This is precisely why computer science should be applied to help solve this problem.

## 1.2 Existing Knowledge and Attempts at Improving Retention in Educational Contexts

Research in this area has isolated some laboratory-tested principles that promote long-term retention [1]:

1. Spaced repetition
2. Practice at retrieval
3. Asking students to learn information in one format and to re-represent it in another format

In this thesis, we will focus on the first principle. We will also discuss how the second principle can be leveraged within this framework. It is our hope that further iterations of this work will find ways to incorporate all three of the aforementioned principles.

There are many flashcard applications, like Quizlet [2], that allow students to practice retrieval at their own pace. Other flashcard applications allow for practice at retrieval while leveraging spaced repetition in order to inform the students' review schedule. Some examples of such systems include Anki, SuperMemo, and Mnemosyne.

Furthermore, there are several classroom techniques that are aimed at promoting long-term retention and transfer. First off, teachers often pose homework, quizzes, and examinations that require students to practice at retrieval. Moreover, teachers often teach in one medium and have students re-represent the information they were taught in a completely different format. An example of this is presenting structured lectures, conveyed in primarily written and auditory formats, to students and asking them to create concept maps or other types of visual representations of the material they were taught.

## 1.3 Overview of Retain

In this paper, we propose a novel method that addresses the aforementioned problems with long-term retention in educational contexts.

The ultimate goal of this system is to take advantage of the psychological spacing effect in order to help students learn course content more efficiently and retain the concepts well past the end of the course. Another goal is to make review and studying more incremental and to preempt cramming behavior before exams.

The system works as follows: when a student learns a new concept in class (this can be any type of class: an on-campus computer science college course, a MOOC physics course, a high school social studies course, etc), they should add the concept to the system as well as a numerical representation of how well they understand the concept. The system then determines when to next remind the student of this concept using a variant of the SM-2 spaced repetition algorithm. Every day, the system chooses a set of concepts to

send to the student for purposes of review. The algorithm for selecting these concepts is the core deliverable of this thesis.

Upon reminding a student of a concept, the student will be asked to provide the system with some information about how well he or she remembered the concept and how well he or she understands the concept now. These two responses will factor into the next iteration of the concept recommendation algorithm.

This system was evaluated with a user study with 10 participants – each of whom had at least one year of teaching experience. The results of the user study overwhelmingly supported the rules upon which Retain was built; on average, 90% of the teachers who participated in the user study chose recommendations produced by Retain over recommendations produced by an existing spaced repetition algorithm or a random concept selection baseline algorithm. Because Retain is built on top of a spaced repetition algorithm, we believe that these results mean that Retain is a system that will improve long-term retention, like many of the existing spaced repetition systems have been able to do, while also not restricting the domain of information to simply concepts with no other conceptual dependencies. In other words, Retain is a general purpose concept recommendation system that promotes long-term retention.

## 1.4 Road Map

In Chapter 2, we will provide an overview of the research on the psychological spacing effect and spaced repetition. In particular, we will focus on a detailed overview of the SM-2 spaced repetition algorithm. We also present an overview of three of the most popular spaced repetition systems: SuperMemo, Anki, and Mnemosyne. These systems, and the features of which they presently comprise, have been influential in the construction of the algorithm and system we propose in this thesis. We also explain how these three systems in particular fail to address the key problem Retain tackles: involving conceptual dependencies in the recommendation process. Chapter 3 walks through the design philosophy and system architecture of Retain. It serves as a primer for the subsequent chapters on the system and its core algorithms. In Chapter 4, we address some important and prerequisite information regarding concept dependency directed acyclic graphs. In Chapter 5, we delve into the details of the algorithm for a basic version of the concept recommendation system. We improve upon this idea by proposing a probabilistic approach to this concept recommendation problem in Chapter 6. In Chapter 7, we discuss implementation specifics with a sample example and some simulated results to make Retain's deliverables more concrete and comprehensible. We also discuss the user study that was used to evaluate the system. This thesis concludes with a discussion of the applications of Retain, some ideas and directions for future work, and a conclusion in Chapter 8.

# Chapter 2

# Related Work

Retain draws from a great deal of work in various areas: both in computer science and experimental psychology. The aim of this chapter is to provide an overview of some related work in order to facilitate a better understanding of Retain.

## 2.1 The Psychological Spacing Effect and Spaced Repetition

This thesis builds upon the ideas, experimental results, and formulas proposed in research on the psychological spacing effect and spaced repetition. Understanding these ideas and the work that has been done in these areas will serve to facilitate the understanding of this thesis as a whole. In this section, we will delve into a concise overview of these two topics.

### 2.1.1 The Psychological Spacing Effect

The psychological spacing effect is a proven phenomenon in learning and cognition. It refers to the findings that assert that studying via mass presentations of information yields substantially inferior results when compared to spaced presentations of the same information. According to Dempster, "this is one of the most remarkable phenomena to emerge from laboratory research on learning" [3].

The psychological spacing effect is a phenomenon in experimental psychology that is both highly dependable and reproducible. It has been validated in virtually every single standard experimental learning paradigm [4] [5] [6].

### 2.1.2 Spaced Repetition

Spaced repetition is a learning technique informed by research on the psychological spacing effect. The general idea is that the process of revisiting a concept – especially as one is about to forget said concept – promotes the long-term retention of the concept. The amount of time that passes before a student is supposed to revisit a concept is important. Different spaced repetition algorithms prescribe different formulae for calculating this time interval. The algorithm detailed in this thesis will use the widely used SM-2 algorithm – originally proposed by the creators of SuperMemo.

There have been several improvements on this algorithm, but SM-2 remains one of the simplest and most effective spaced repetition algorithms in practice. It is also an algorithm that is used by popular and effective applications that employ spaced repetition like Anki and Mnemosyne. Both of these applications elected to use the SM-2 algorithm over other spaced repetition algorithms, including the later SM-iterations, due to its efficacy and simplicity.

### 2.1.3 SM-2 Algorithm

Below is the interval calculation algorithm used in the Retain concept recommendation system; this algorithm is called SM-2 [7].

1. Split knowledge into small items
2. Associate an E-Factor (initially set to 2.5) with each item. The E-Factor reflects the easiness of memorizing an item.
3. Repeat items using the following intervals:

$$I(n) \leftarrow \begin{cases} 1 & \text{if } n = 1 \\ 6 & \text{if } n = 2 \\ I(n-1) * EF & \text{if } n > 2 \end{cases}$$

   where $I(n)$ is the inter-repetition interval after the concept has been repeated n times. $I(n)$ is measured in days. $EF$ is the E-Factor of a given item (rounded to the nearest integer).
4. After each repetition, gather an assessment of how well the user was able to remember the concept on a 0-5 scale (where 0 indicates a complete blackout and 5 represents perfect recall).
5. After each repetition, update the concept's E-Factor as follows:

$$EF' \leftarrow EF + (0.1 - (5 - q) * (0.08 + (5 - q) * 0.02))$$

   where: $EF'$ is the new value of the E-Factor, $EF$ the is old value of the E-Factor, and $q$ is the quality of the response on the 0-5 grade scale. If $EF$ is less than 1.3, let $EF$ be 1.3.
6. If the quality of response, $q$, was lower than 3, start repetitions for the item from the beginning without changing the E-Factor (i.e. use intervals I(1), I(2) etc. as if the item was memorized anew).
7. After each repetition session on a given day, repeat all items that scored below 4 in the quality assessment. Continue the repetitions until all of these items score at least 4 in the quality assessment.

It is important to note in that in the Retain system, the SM-2 algorithm does not completely dictate when a student is reminded of a concept. The SM-2 algorithm generates a suggestion for the next time the user should revisit the concept, but the Retain algorithm does some additional work beyond that.

## 2.2 Existing Spaced Repetition Systems

Retain was influenced by existing systems in the realm of spaced repetition software. There are several such software systems – we will discuss three of the most popular and widely used systems in this chapter. All three of these systems help users learn and manage information in the form of flashcards. They also allow users to create their own flashcards with rich content. Users also have access to flashcard decks made by other users. Each of these applications presents users with flashcards and factors in user feedback, regarding how comfortable the user is with the concept or how easily the user was able to retain the concept, when using a spaced repetition algorithm to determine when to review the concept next. Lastly, all three of these systems allow for the generation of cloze deletion cards (i.e. automatically generating flashcards from a block of text. The flashcards will have some part of the text omitted, and it is the student's job to try to fill in the blanks).

We will discuss the merits and details of each of these systems – we will also highlight how each system fails to solve what the Retain system solves.

### 2.2.1 SuperMemo

A lot of the work described in this paper is based off of the efforts and research conducted by SuperMemo [8]. SuperMemo is perhaps one of the most popular and influential systems within the realm of computer-assisted learning via spaced repetition. A lot of systems built their algorithms off of the early SuperMemo algorithms (particularly the SM-2 algorithm); we will mention some of these systems in the subsequent sections of this chapter.

SuperMemo is a flashcard system that employs spaced repetition. It has evolved a great deal through the years. Presently, SuperMemo uses an algorithm called SM-16. The most recent version of SuperMemo boasts incremental reading, incremental learning, visual learning, knowledge tree categorization infrastructure, and the ability to assign priorities to concepts.

SuperMemo has one feature that is nominally close to the ideas presented in this paper – it allows learners to arrange their concepts within a knowledge tree. It is important to note that SuperMemo explicitly mentions that the structure of this knowledge tree does not affect the learning process [9]. The tree simply serves as a categorization mechanism for concepts; it enables users to easily organize their concepts by topic and granularity. The tree does not explicitly require that users organize data in a manner that represents conceptual learning dependencies, and the algorithm does not leverage this structure when making concept recommendations.

### 2.2.2 Anki

Anki [10] is a flashcard system that employs spaced repetition techniques in order to help students learn concepts more expediently. Anki's algorithm was built on top of SuperMemo's SM-2 algorithm. The author of Anki asserts that later versions of SuperMemo (specifically SM-3 - SM-5) are more susceptible to erroneous

scheduling. Anki has modified SM-2 by also considering urgency and user priorities when determining which concepts to suggest to students. Anki's underlying algorithm takes care to make sure that related cards are not suggested to the user in a short spacing.

Anki is an open source project and has a well-populated library of add-ons. One add-on allows for the hierarchical tagging of concepts. This functionality solely seems to be used for categorization purposes; the hierarchical dependencies managed by this add-on do not affect the interval calculations.

### 2.2.3 Mnemosyne

Mnemosyne [11] is a spaced repetition software system that builds upon the SM-2 SuperMemo algorithm.

Students have full control over the cards or the card types they want to study. The system allow students to easily focus on a subset of their cards in a particular session. The system also provides statistics that address learning progress. Mnemosyne comes with a number of plugins out of the box. One such plugin is a cramming scheduler – this plugin allows students to drill through a set of cards before a particular deadline, say an exam, without affecting the default scheduling mechanisms

## 2.3 Problems with Using Spaced Repetition for Non-Atomic Pieces of Knowledge

Spaced repetition has been successfully employed in flashcard applications. Existing spaced repetition algorithms are perfectly suited for atomic pieces of information. Here, we define *atomic* as a concept that's theoretically self-contained and does not have any major conceptual dependencies. An example of an atomic concept is the Spanish word for house, "casa". In order to learn and revisit the mapping of "house" to "casa", we do not need to worry about our comprehension of any other concepts. Now, let's think about the concept of a "chemical reaction". In order to understand chemical reactions, we need to have a passable understanding of several things, like energy, atoms, etc. It wouldn't make much sense to revisit/re-learn "chemical reactions" if we didn't have a decent grasp of those aforementioned concepts. This is where exclusively relying on existing spaced repetition algorithms, like SM-2, for our task breaks down. It is clear that we need to add some additional algorithmic infrastructure before we can use the interval calculation formulae prescribed by SM-2 and the like.

# Chapter 3

# Architecture and Design Philosophy of Retain

In this chapter, we will discuss a high-level overview of Retain as well as some of the motivations behind the concept recommendation algorithm.

## 3.1 Important Definitions

We'll need to define a few critical terms before we can discuss the system. These concepts will be explained more formally and in greater detail in future chapters:

1. A *concept dependency graph* is a directed acyclic graph that encodes dependencies between concepts.
2. Each concept has a *next review time* with which it is associated. This time is determined by the SM-2 spaced repetition algorithm. It tells us when the concept should be sent to the user next.
3. A *timely concept* is a concept whose *next review time* has passed.
4. A concept $b$ is *ancestral* to another concept, $a$, if there is a directed path in the concept dependency graph from $a$ to $b$.
5. Each concept has a *comprehension score (C-Score)*; this is a numerical value that represents the student's comprehension of the concept.

## 3.2 System Requirements and Deliverables

Each student has his or her own instance of the system. Each instance takes two inputs: a concept dependency graph for the course (which we'll discuss more in Chapter 4) and a number, $N$, that represents the number of concepts the student would like to revisit and study each day.

Every day, the system's algorithm is executed, and it recommends up to $N$ concepts for the student to study. These are concepts that the student has already learned; they are recommended to the student for purposes of revisitation and review. The algorithm for how these concepts are chosen will be discussed in Chapter 5.

Upon receiving the concept recommendations for the day, the student is expected to review these concepts and, after doing so, provide the system with some feedback on how how well they retained the concept and how well they understand the concept after reviewing it. This information feeds the next run of the concept recommendation algorithm.

## 3.3 High-level System Architecture

Below is a workflow diagram that visually explains the overall workflow for a particular student on a particular day. This workflow diagram abstracts away a lot of nuances and implementation-level details; we will cover a lot of these details as we delve into future chapters.

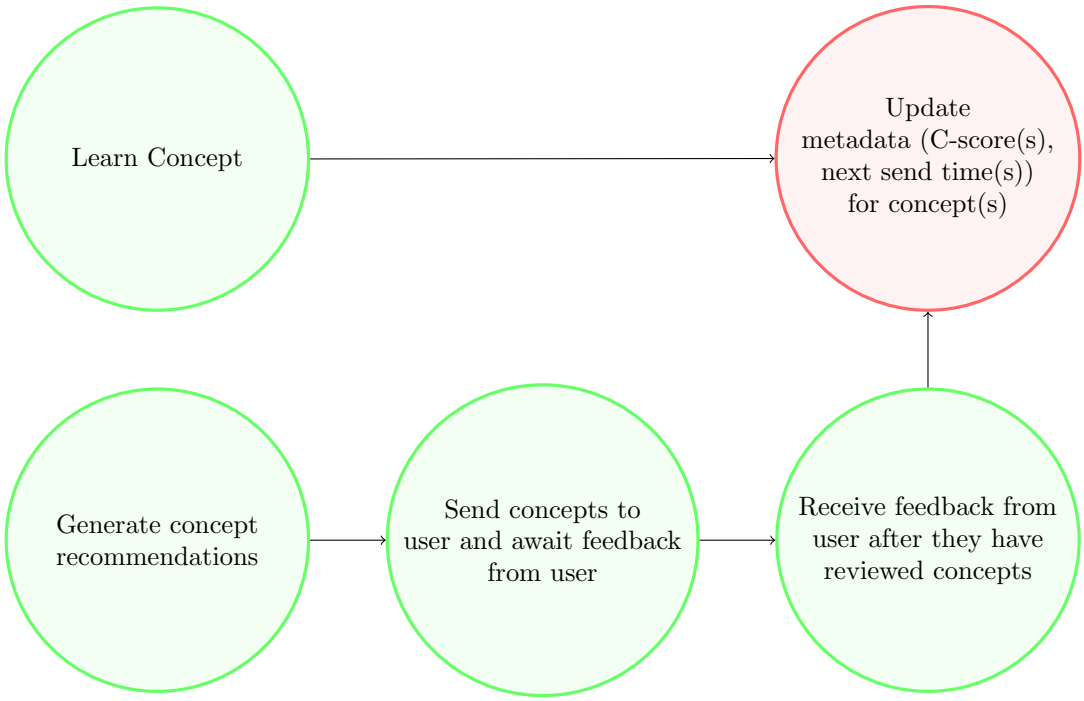



Figure 3.1: Retain workflow diagram

## 3.4 Generating Concept Recommendations

Now that we have an idea of the overall inputs and outputs, workflow, and goals of Retain, we can start talking about perhaps its most important component: the concept recommendation algorithm. Chapters 5 and 6 are devoted to this topic; but before we discuss specifics and formalism, let's first discuss some of the intuition, motivations, and design decisions behind the construction of Retain's concept recommendation algorithm.

### 3.4.1 Selecting a Spaced Repetition Algorithm as a Starting Point

The Retain concept recommendation algorithm uses the concept suggestions generated by a spaced repetition algorithm as a starting point. A major decision point was choosing which spaced repetition algorithm to use to generate these seed concepts. The thought process was that since we were deviating from the exact intervals prescribed by the spaced repetition algorithm in the interest of yielding to other concerns, like conceptual dependencies, it was more important to choose a spaced repetition algorithm that was simple, extensively tested, and experimentally validated than to choose the most sophisticated algorithm in existence. SM-2 is an algorithm that is still used in some of the most popular spaced repetition systems; it was introduced several years ago and it has stood the test of time. Thus, we chose to use SM-2 as a starting point for our concept recommendation algorithm.

### 3.4.2 Motivations

There are several abstract motivations that guided the rule and algorithm generation process for this recommendation system.

1. A student should only be reminded of a concept once he or she has learned it (once it has been taught in class, once the student has read a chapter on it, etc.).
2. It does not make sense to blindly send timely concepts to the student if the timely concepts have ancestors with low comprehension scores. We should send those ancestral concepts first.
3. Because we want to adhere to the spaced repetition principle as closely as possible, we should factor in the prioritization of concepts with the earliest next review times.
4. When trying to decide which concept to send to the user from amongst a group of concepts, we should prioritize the concepts that are more fundamental (or to be precise, ancestral) to the largest number of timely concepts (or concepts with low comprehension scores that are ancestors to timely concepts). The reason we care about the latter is best explained using the following graphic:

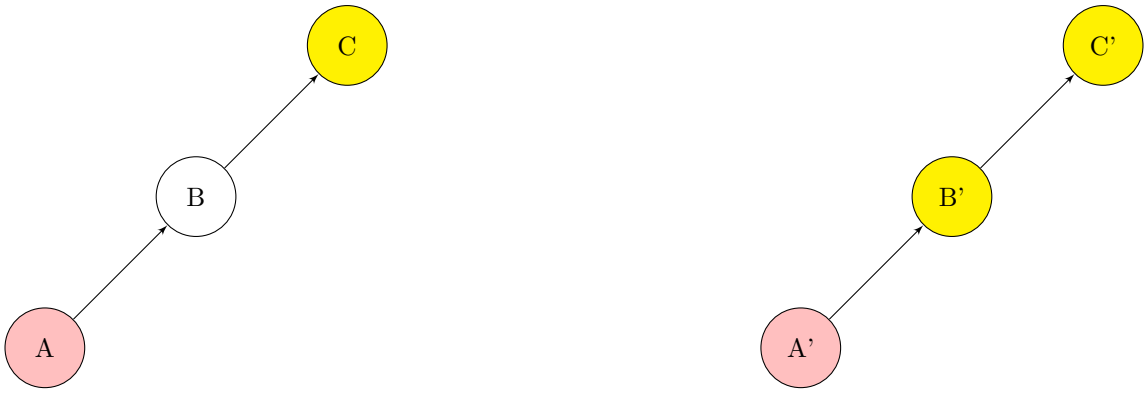



Figure 3.2: Yellow nodes represent concepts with low C-scores. Pink nodes represent timely concepts. White nodes are nodes without any noteworthy properties.

We know that it's going to take more time to fulfill the prerequisites for timely concept A' because we have more concepts to remind the student of; therefore, we want to prioritize A' over A.

5. If we're going to make multiple recommendations in a day, say we want to recommend three concepts in a day, we should make sure to factor in the idea that the student will review the concepts in the order they were recommended to him or her. As such, once the student has reviewed the first concept and updated their comprehension of the topic, the remaining recommendations could become outdated. For example, if we determine that concepts A, B, and C are the concepts the user would gain the most from revisiting and should therefore be recommended, we should note that it's possible that once the student reviews concept A and updates the respective C-Score, if we were to re-rank the concepts by utility, it is possible that B and C wouldn't be the next two most highly ranked concepts. In this case, we would want to base our recommendations off of the newest available data rather than outdated data.

### 3.4.3 Concept Recommendation Algorithms

The aforementioned motivations serve as a foundation for the two algorithms we propose in the coming chapters. In Chapter 5, we detail a deterministic basic concept recommendation algorithm. We also propose a probabilistic algorithm that improves upon the flaws of the basic algorithm in Chapter 6.

# Chapter 4

# Concept Dependency Directed Acyclic Graphs

Retain's algorithm depends heavily on concept dependency directed acyclic graphs. In order to support non-atomic concepts, it's necessary for us to have information about the structure of and relationship between concepts in the course. This is so we can make sure that a concept's ancestors have been properly understood before asking the student to study the concept itself.

## 4.1 Generation of Concept Dependency DAGs

The current iteration of this system relies on a manually generated concept dependency DAG. Automatically generating concept dependency DAGs is an active research area, and the hope is that in the near future, the graph generation can be completely automated. Retain assumes that a concept dependency DAG is provided as an input. This DAG contains all of the concepts (and their respective relationships) that are covered throughout the span of the course.

### 4.1.1 Prior Works in the Realm of Automatic Concept Graph Generation

As mentioned above, the problem of automatically generating a concept graph is an active research area. There have been several contributions to this field in the realm of mining the dependencies between concepts from text (Liu et al.) [12], the measurement of prerequiste relationships between concepts (given documents that represent each concept) (Liang et al.) [13], and extracting heirarchical information about concepts from textbooks (Wang et al.) [14]

## 4.2 Example

Since the remainder of this paper will use terminology and diagrams related to concept dependency DAGs, it would be helpful to go over the structure and relationships encoded in these DAGs. This is an example of a concept dependency DAG with only six concepts. The edges carry the following meaning: the source node of the directed edge is dependent on the destination node of the directed edge. For instance, in this example, Derivatives depends on Arithmetic. This means that it is important that a student masters arithmetic before moving onto revisiting and internalizing derivatives.

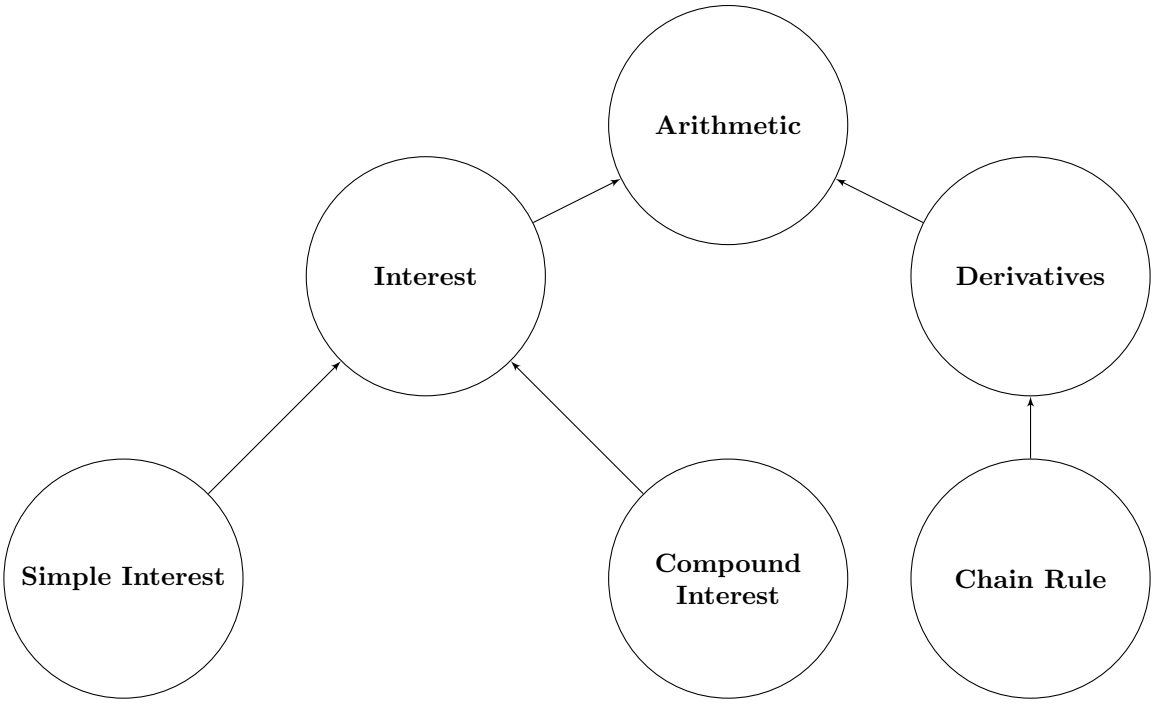



Figure 4.1: Example concept dependency DAG

# Chapter 5

# Basic Concept Recommendation Algorithm

In this chapter, we will discuss the center-point of Retain: the basic concept recommendation algorithm. This is the algorithm that determines which concepts to recommend to a particular student on a given day. This chapter draws from knowledge we've discussed in all of the previous chapters. We'll start with an overview of some necessary definitions. Next, we'll discuss some of the rules upon which the algorithm hinges. We'll then proceed to formalize the algorithm itself. Finally, we'll discuss some of the drawbacks of this algorithm.

## 5.1 Definitions

First, we need to define several concepts we need to properly formalize the basic concept recommendation algorithm.

1. The concept recommendation algorithm detailed in this paper is dependent on some evaluation metrics that provide insight into the progress and understanding exhibited by a student. For the time being, these metrics will be wholly determined by self-assessments. In the future, it would be a good idea to involve a more objective evaluation mechanism to determine these metrics. These self-evaluations will be performed using a standard Likert scale (scale from 0-5).

   (a) The *Retention Score (R-Score)* is a self-reported measure of the how well and easily the student was able to remember the concept and its associated details as they are reminded of the concept and prior to reviewing it.

   (b) The *Comprehension Score (C-Score)* is a self-reported measure of how well the student understands the concept and its associated details as they are reminded of the concept and after reviewing it.

2. Each concept has a *next review time* with which it is associated. This time is determined by the spaced repetition algorithm. It tells us when the concept should be sent to the user next.

3. A *timely concept* is a concept whose *next review time* has passed.

4. Let's call the distance from concept $a$ to concept $b$, $d_{ab}$. This is the length of the directed path from $a$ to $b$. If there is no directed path from $a$ to $b$, $d_{ab} = -1$. Lastly, the distance from concept $a$ to itself, $d_{aa} = 0$.

5. A concept $a$ is *ancestral* to another concept, $b$, if $d_{ba} \geq 1$.

6. If a $C - Score \leq 2$, it is considered to be a low $C - Score$

7. Suppose $T$ is the set of timely concepts, $lowC$ is the set of concepts with low C-Scores, and $ancestors$ is a set of all the concepts that are ancestral to concepts in $T$.

$$activeConcepts = T \cup (lowC \cap ancestors)$$

8. *timeRank* is a measure of how early a concept's $nextReviewTime$ is; its range is $(0, 1)$ because, by design, $nextReviewTime > 0$. Earlier *next send times* have *timeRanks* closer to 1.

$$timeRank(j) = 1 - \frac{nextReviewTime_j}{\max\limits_{c \in activeConcepts} nextReviewTime_c + 1}$$

9. The *rank* of a concept is a measure of how fundamental the concept is. The intuition behind this definition is as follows: the higher up in a dependency tree a concept is and the more dependents a concept has, the more fundamental a concept is. If a concept is fundamental, it should be prioritized. Therefore, the rank of such a concept should be higher as rank is used to decide which concepts would be the best choices to send to a student (based on helpfulness and efficiency).

   The *rank* of concept $j$ is defined as follows:

$$rank(j) = timeRank(j) + \sum_{i \in activeConcepts} \max(d_{ij}, 0)$$

   $timeRank(j)$ is used as a tie-breaker for ranks. Sometimes, aggregate distance from active concepts isn't enough to determine a ranking without ties. In these cases, we attempt to break ties by using each concept's next review time. If a concept has a sooner (smaller) next review time, we want to make its rank larger. Note that it is still possible to have ties if two concepts have identical next review times and aggregate distances from active concepts.

## 5.2 Rules for Concept Recommendations

1. Only remind a student of a concept once the system has received an explicit cue indicating that the student has been taught the concept.

2. Use $rank(x)$ as the primary measure for selecting concepts to recommend. The higher the *rank* the more the concept should be prioritized.

3. Once a concept is selected to send to the student, make the strong assumption that that concept has a perfect comprehension score, and select the next concept to send for today by re-ranking concepts based on that assumption.

## 5.3 Formalizing the Algorithm

In this section, we will formalize an algorithm that takes into account each of the rules mentioned in the previous section. This algorithm operates in discrete time steps, $t \in \mathbb{Z}_{>0}$. Each student is associated with an instance of the algorithm. We describe the algorithm instance for a singular student in Section 5.3.2.

### 5.3.1 Set-up

First, we need to do some one-time setup. We want to generate a number of mappings. For each concept, $a$, we want to store the distance in the concept DAG from $a$ to every other concept in the graph. We can do this by running the *Floyd-Warshall* algorithm [15][16] for all pairs shortest path on the concept dependency DAG. This will give us the distance from every concept to every other concept in the DAG. The length of the directed path from concept $a$ to concept $b$ is $d_{ab}$. If there is no directed path from $a$ to $b$, Floyd-Warshall will set $d_{ab}$ to $\infty$. In these cases, we want to override Floyd-Warshall's output in the following manner: $d_{ab} = -1$.

### 5.3.2 Algorithm for a Singular Student

1. We maintain two key pieces of data `Cscores` and `nextReviewTimes`. The former maps each concept to a comprehension score; the latter maps each concept to a time value. Each of these data structures is updated over time. `Cscores[c]` refers to the `C-Score` associated with concept $c$. Likewise, `nextReviewTimes[c]` refers to the next review time associated with $c$. All entries in both `Cscores` and `nextReviewTimes` are initially set to *null*.

2. When a student learns a new concept, $c$, they are instructed to alert the system. The student is expected to provide the system with the concept name and the respective $C - Score$. We update our records with this information. We also use the SM-2 algorithm to determine the next time the student should revisit $c$; let's call this time $t'$. We set `nextReviewTimes`$[c] = t'$.

   We formalize this logic in the function `initializeConcept(concept, Cscore)` – which is defined below. But first, we define the following helper function to assist in defining this function as well as a forthcoming function:

   > `generateNextSendTime(concept, newRScore)`: using the SM-2 algorithm, which is explained in detail in Section 2.1.3, generate a next review time for the concept where $q$, the parameter in SM-2 that measures the quality of the student's response, is `newRScore`.

---

**Function** *initializeConcept (concept, Cscore)*
  lowestRscore ← 0;
  nextReviewTime ← generateNextReviewTime(concept, lowestRscore);
  nextReviewTimes[concept] ← nextReviewTime;
  Cscores[concept] ← Cscore;

---

3. At each time step, $t$, the algorithm determines which concepts to recommend to the student.

4. In this step, we will incrementally define the logic for generating and sending concept recommendations at time $t$.

   Below is the definition for `recommendNextConcept()`. This function decides which singular concept to recommend next. If there are no valid concepts left to recommend – i.e. no concepts that satisfy the aforementioned rules – this function will return *null*.

   Once the concept has been determined, it is sent to the student (the medium of communication is relatively unimportant; a proposed medium for sending these concepts is e-mail). We also make the strong assumption that the student understands the recommended concept perfectly, and we update the concept's C-Score accordingly.

```
Function recommendNextConcept ()
    perfectScore ← 5; // The highest possible C-Score value
    conceptRankMap ← empty map;
    for concept in activeConcepts do
        conceptRankMap[concept] ← rank(concept);
    recommendation ← null;
    greatestRank ← −1;
    for < concept, rank > in conceptRankMap do
        if rank > greatestRank then
            recommendation ← concept;
            greatestRank ← rank;
    // Make the strong assumption that the the user understands the recommended
       concept perfectly
    Cscores[recommendation] ← perfectScore;
    if recommendation is not null then
        sendRecommendationsToUser(recommendation);
```

   To tie everything together, below is the definition for `recommendConcepts(N)`. This function determines and sends up to $N$ concepts that should be recommended at the present time, $t$.

```
Function recommendConcepts (N)
    for i from 1...N do
        recommendNextConcept();
```

5. The student will study the concepts they receive in the order in which they received them. After finishing revising a concept, the student must update the system with information that indicates which concept they revised, how well they understand the concept now (measured by the C-Score, *newCscore*) and how easily they were able to remember the concept (measured by the R-Score, *newRscore*). Below is the function that is executed every time a user alerts the system that they have revisited a concept.

```
Function recieveUserFeedback(concept, newCscore, newRscore)
    Cscores[concept] = newCscore;
    newNextReviewTime ← generateNextReviewTime(concept, newRscore);
    nextReviewTimes[concept] ← newNextReviewTime;
```

## 5.4 Drawbacks of the Algorithm

1. It is possible that certain concepts could get continuously delayed (and, in the worst case, never get reviewed) due to the algorithm's requirement that concepts can only be reviewed if all of their ancestors have been properly understood. In these cases, the intervals between successive revisions of concepts could grow to be too large – rendering the spaced review of concepts to be ineffective.

2. Under this algorithm, it is possible to get "stuck" – to continuously be reminded of the same few concepts over and over due to the deterministic nature of the algorithm and the strict prerequisites for revisiting a concept. In the next chapter, we will propose an improvement upon this algorithm that will address this flaw.

# Chapter 6

# Building off of the Basic Algorithm

This chapter will contain some ideas and an algorithm that addresses some of the weaknesses of Retain's basic concept recommendation algorithm. We will go through the intuition for and details of a probabilistic approach to concept recommendations that is heavily founded on the principles and ideas discussed in Chapter 5.

## 6.1 Objective

We want to avoid getting "stuck" and recommending the same concepts to the student over and over again. For instance, suppose a particularly difficult group of concepts have high ranks and keep getting recommended by the system. Suppose, in this scenario, that the student is having difficulty comprehending these concepts, and as such, regardless of the number of times he or she reviews these concepts, cannot get past them. While it makes sense for the system to remind the student of these concepts often, it would be ideal if these weren't the only concepts the student was reminded of – as this could be a waste of time, counterproductive, or frustrating to the student. Allowing some reprieve from repeated review sessions involving exclusively difficult concepts could be positive and encouraging for a student and could also help them make progress on a wider range of material.

## 6.2 Introducing Probabilistic Selection

In order to address the aforementioned objectives, we introduce a modification to the algorithm detailed in the previous chapter. This modification was inspired by the second drawback discussed in Section 5.4. A deterministic algorithm can have negative consequences in the context of this concept recommendation system. There is a certain level of uncertainty that is involved in each of the ranking calculations and as such, it is fitting to involve some probabilistic selection when choosing amongst the ranked concepts. Furthermore, as expressed in Section 6.1, probabilistic selection can help us avoid getting stuck with repeated recommendations of difficult concepts, and it can also introduce some chance of reprieve between reviewing exclusively difficult concepts.

### 6.2.1 Concept Recommendation Algorithm

The framework for this algorithm is very similar to that of the algorithm formalized in Section 5.3.2. The only difference is the manner in which the next concept to recommend is selected. Below, we detail the new algorithm for choosing the next concept to recommend:

Suppose *validConcepts* is the set of concepts that have zero ancestral concepts with low C-Scores, and *allConcepts* is the set of all concepts in the concept dependency DAG.

We define a new ranking function, $R(x)$, in the following way:

$$R(concept) = \begin{cases} rank(concept) & \text{If concept is in } validConcepts \\ 0 & \text{Otherwise} \end{cases}$$

We define a multinomial distribution over the set of all concepts. The probability weights of this distribution are specified by the vector $\beta$, in which the probability of choosing the concept $i \in allConcepts$ is defined in the following manner:

$$P(X = i) = \frac{R(i)}{\sum_{c \in allConcepts} R(c)}$$

When choosing the next concept to recommend, $r$, we sample from the aforementioned distribution:

$$r \sim Multi(\beta)$$

### 6.2.2 References

The practice of using probabilistic selection for recommendation systems rather than always choosing the highest ranked items in a deterministic fashion is a common theme in recommendation systems. One example is the Shop Recommendation System proposed in Style in the Long Tail: Discovering Unique Interests with Latent Variable Models in Large Scale Social E-commerce (Hu et al.) [17]

# Chapter 7

# Implementation, Results, and Evaluation

In this chapter, we will present details about the implementation of Retain and some sample, simulated results in order to facilitate understanding of the types of results Retain produces. We will also discuss how we evaluated the recommendations produced by Retain.

## 7.1 Implementation

Here, we will discuss some of the specifics related to the implementation of the algorithm discussed in Chapter 5. We will also walk through the details of a sample example created solely to facilitate understanding of how the system is supposed to operate in totality.

### 7.1.1 Source Code

The source code for this system can be found at: https://github.com/Shilpaks/retain

This repository contains all the back-end endpoints necessary to deploy the algorithm discussed in this paper in the context of any course. The repository also contains documentation, tests, and an example web application.

### 7.1.2 Implementation Details

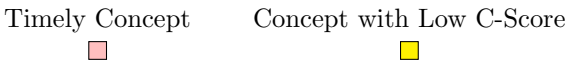
The source code was entirely written in Python. The example does contain some front-end infrastructure; however, this was purposely contained to the example and is not part of the system as a whole. After some experimentation and iteration, it became clear that it is somewhat unnatural to have a separate interface for the functionality provided by Retain. It is tedious for students to interact with a primary online course platform like Coursera, Scholar, Compass, etc. while also maintaining records of their progress and retention on a separate website. For this reason, it would be more helpful if this back-end code was agnostic to user interfaces – allowing anyone to embed this functionality within existing or new websites.

### 7.1.3 Sample Example

The sample example (which can be found in this project's GitHub repository mentioned in Section 7.1.1) consists of a simple Flask application that uses the Retain API. There is a view in which the student can add a new concept that they've recently learned to the system (as well as the corresponding comprehension score). There is a second view in which a student can tell the system that they've revisited a particular concept and also provide the corresponding comprehension and retention scores. The UIs for these views can be found in Appendix A.

## 7.2 Simulated Results

To provide a look into the workings of both the basic and probabilistic concept recommendation algorithms and the types of results they generate, we'll take a look at some simulated results. The sample graphs are color coded using the following legend:




### 7.2.1 Basic Concept Selection Algorithm

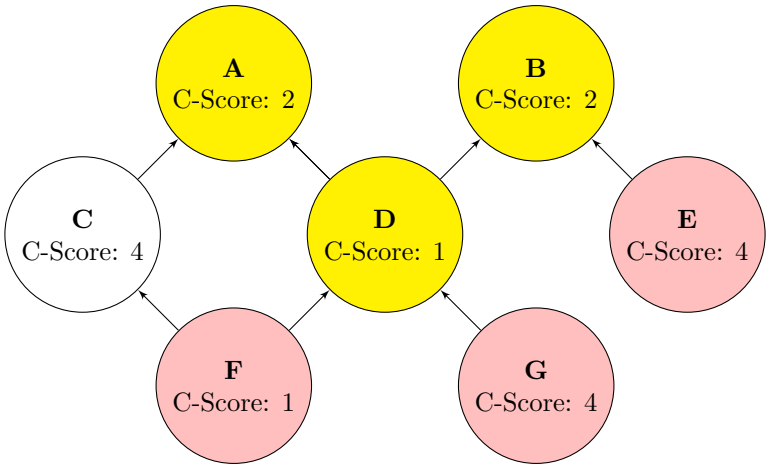



Figure 7.1: Top three recommended concepts (in order of priority) by the basic concept recommendation algorithm: B, A, D

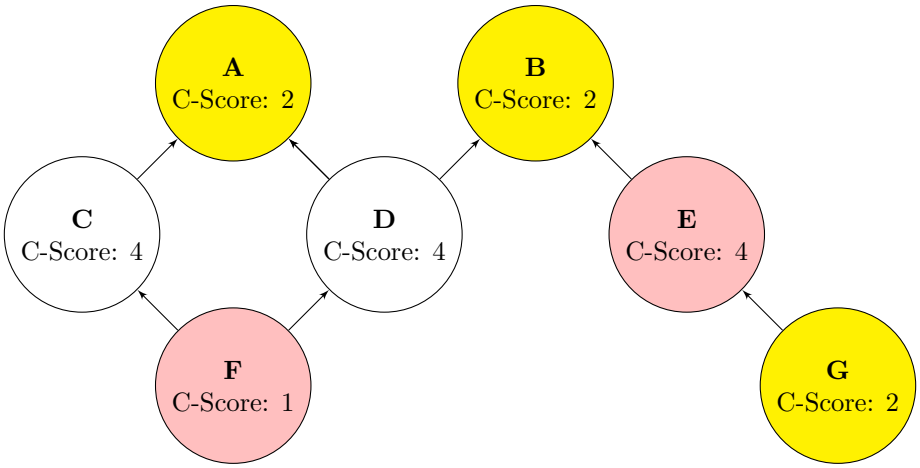



Figure 7.2: Assumption: E's next review time is sooner than F's. Top three recommended concepts (in order of priority): B, A, E.

### 7.2.2 Probabilistic Concept Recommendation Algorithm

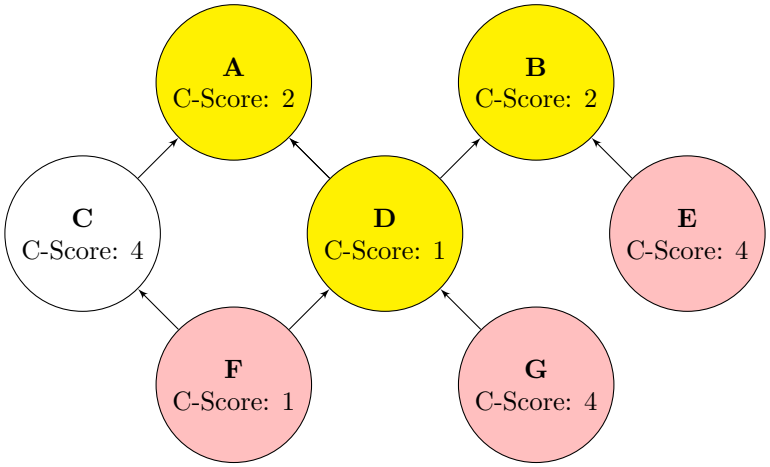



Figure 7.3: Top three concepts recommended by the probabilistic concept recommendation algorithm: B, A, E

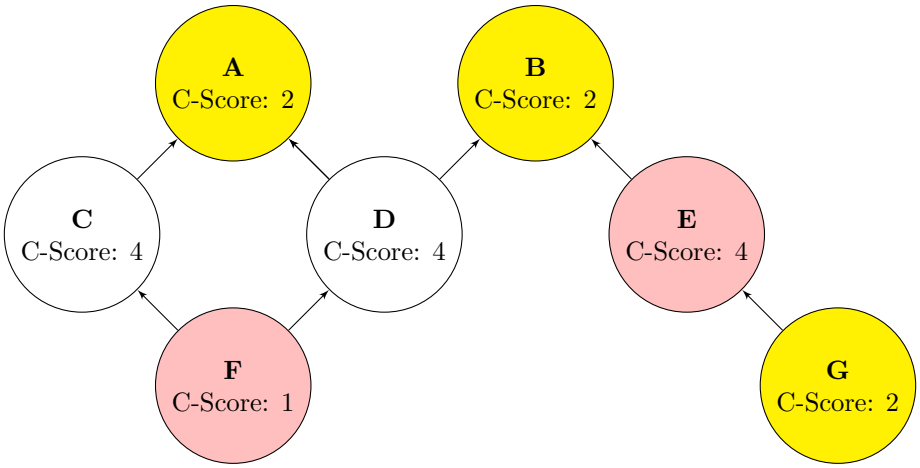



Figure 7.4: Top three concepts recommended by the probabilistic concept recommendation algorithm (in order of priority): B, E, A

## 7.3 Evaluation

The goals of the evaluation are two-fold:

1. Show that involving concept dependencies in spaced repetition recommendation tasks is important.
2. Show that the algorithm's results and rules can be validated by individuals with teaching experience.

### 7.3.1 Difficulties with Evaluation of Concept Recommendation Algorithms

Finding a way to evaluate the concept recommendation algorithms detailed in this paper, without going through with a long-term classroom evaluation of the system and analyzing pre-test and post-test results, is non-trivial. A purely quantitative approach to evaluation (i.e. simulation) is infeasible due to the reliance of the algorithm on user evaluation and input. For this reason, we decided to design and implement a user study in order to test the fundamental rules upon which both algorithms proposed in the earlier chapters depend.

### 7.3.2 User Study Design

The user study contains three scenarios. Each scenario aims to subtly test the rules laid out in Section 5.2. They also make a veiled attempt to ask participants to choose between results produced by the Retain algorithm, SM-2, and a random concept selection baseline. Participants are asked to read a story-like scenario, consult the provided concept dependency graphs that represent the scenario, and answer a series of multiple choice questions. After every question, the participants are asked to write a brief statement justifying their choice. The user study questionnaire can be found in its entirety in Appendix B.

### 7.3.3 User Study Results

For the user study, we decided to test the results produced by Retain's basic concept recommendation algorithm. Since we were really trying to test whether the rules upon which both the basic and probabilistic algorithms were built could be validated by experienced teaching professionals, we decided to not let the non-determinism associated with the probabilistic algorithm skew or bias the results. We fully expected that any positive evaluation results produced from this user study could be generalized to support the probabilistic concept recommendation algorithm.

The teaching professionals who participated in the user study overwhelmingly favored Retain's algorithm. The final question of every scenario asked the user to choose amongst the recommendations produced by the SM-2 algorithm, the Retain algorithm, and random concept selection. 10 individuals with at least one year of teaching experience participated in the study. Across all three scenarios, the recommendations produced by Retain were chosen **90%** of the time.

#### Quantitative Evaluation

| | Scenario 1 | Scenario 2 | Scenario 3 | Average Over All Scenarios |
|---|---|---|---|---|
| $P$ | **80%** | **100%** | **90%** | **90%** |

Table 7.1: Quantitative results from the user study.

where $P$ is the percentage of respondents who, when asked to choose which recommendations they would prefer to suggest to the student, chose the recommendations produced by Retain (over saying "I don't know/No preference" or choosing a recommendation produced by either SM-2 or random concept selection).

#### Qualitative Feedback

We asked the participants of the user study to provide a written rationale for each of the choices they made on the multiple choice questions. These responses shed light on the types of things teachers tend to care about when recommending concepts to students or just teaching in general.

A lot of the written feedback supported Retain's recommendation rules. The majority of teachers surveyed agreed that it's important to address fundamental concepts with low comprehension scores before addressing timely concepts.

Of the minority of teachers who favored concepts and groups of concepts that weren't recommended by Retain, the general vein of thought was that sometimes, Retain recommended concepts that were too fundamental. Some teachers thought revisiting these concepts wouldn't be productive as they are so far removed from the concepts recommended by SM-2.

One particularly interesting trend that we found from reading the responses was that teachers wanted some degree of relatedness between the concepts selected in a particular batch of recommendations. Some

teachers asserted that it's important that the concepts a student is told to review on a particular day are not too disjoint and that they are in some way related to one another. This was helpful feedback, and future iterations of Retain will factor this into the recommendation algorithm.

# Chapter 8

# Applications, Future Work and Conclusion

In this chapter, we will discuss the various different ways Retain (or the meta-data it collects) can be used in practical situations or to further and bolster research in the domain of education. We will also discuss the next steps that should be taken to improve Retain and innovate on top of its foundation. Lastly, we will conclude with some summative ideas and observations.

## 8.1 Applications

This system was designed with courses in mind. Of course, it is possible for this system to be used for purposes of independent study, but our true vision for this system is for it to be used by an entire class. The concept dependency DAG would be provided by the professor of this course, and each student would interact with his or her own instance of this system throughout the span of the course. The primary goal of this system is to help students increase their long-term retention of the topics covered in the course. A secondary goal is to distribute the learning and revision required for the course intelligently and lessen the burden on the student at semester pressure points like exams and quizzes.

Some systems we could envision this this system being integrated with include: Coursera, Scholar, Moodle, and Compass2g. It is also possible for courses and independent learners to use this system and its standalone web application without any other learning platform integration. This is more tedious, and it requires the student to keep track of more websites, which isn't a good thing, but it is definitely doable if a student prefers to go this route.

Additionally, after a sufficient amount of usage, the system will have, for each user and each concept, a measure of difficulty. This is the E-Factor that's maintained and incrementally updated by SM-2. For each concept, all of the students' E-Factors can be aggregated and condensed into one representative number per concept (by taking the median or mean, for instance).

## 8.2 Future Work

The ideas proposed and formalized by Retain leave lots of room for improvement and innovation. In many ways, the foundation for future work is one of Retain's biggest contributions. In this section, we will discuss two types of work that should be done in the future: a more in-depth evaluation and innovation and

improvements on top of the two algorithms detailed in this paper.

### 8.2.1 Long-Term Classroom Evaluation Plan

Because this system's goal is to boost long-term retention, the best way to test this system and the efficacy of the underlying algorithms is to actually deploy this system in an actual course. Because it wasn't feasible to complete this evaluation for the sake of this thesis, we will lay out a plan for proper evaluation of this system below.

For the purposes of specificity, the following evaluation plan assumes successful integration with a learning platform called Scholar. However, this plan is perfectly feasible for other integrations.

1. Choose a course that uses the Scholar platform and has had a previous offering that does not differ too greatly from the present offering. We want this because it would be helpful to compare the performances of students in each semester (compare the performances of students who used Retain against the performances of students who didn't). It is important that this course also has at least 2-3 objective assessments throughout the course of the semester; it is particularly important that the course has a comprehensive assessment at the end of the term. This is because we want to be able to evaluate how well the students have been able to retain and internalize information in the long-term.
2. Compare the midterm and final results between the two sections (the section that didn't use Retain verses the section that did).
3. Determine whether the difference between the two sections' scores is statistically significant.
4. At the end of the term, circulate a questionnaire amongst the class for the purposes of qualitative evaluation.

### 8.2.2 Future Modifications to Concept Recommendation Algorithm

1. The qualitative feedback collected during the user study highlighted an interesting potential fault that should be addressed in future iterations of Retain. Suppose we have two concepts, $a$ and $b$. Suppose $a$ is ancestral to $b$. There might be a time at which, due to delays between collecting updated C-Scores, $a$ might have a comprehension score that is much lower than that of $b$. From a teacher's perspective, this does not make much sense. How can we act on information that tells us that $a$ has been poorly understood, and $b$ is fairly well understood, yet $b$ conceptually depends on $a$? As a starting point, we can think about bounding the comprehension scores of each concept by the lowest ancestral comprehension score.
2. In the future, we should try to consider how "related" concepts in the same batch of recommendations are. As mentioned earlier, some of the teachers who participated in the user study noted that a key consideration, when judging the strength of a set of concept recommendations, is how related the concepts in the set are. The more related, the better.

## 8.3 Conclusion

Overall, the teachers who participated in the user study preferred the results of the Retain concept recommendation system to results produced by both a traditional spaced repetition algorithm, SM-2, and a random concept selection algorithm. Therefore, these teaching professionals agreed that Retain is an improvement over simply relying on spaced repetition to select which concepts to review next. There is still a lot of work to be done going forward. We collected a number of helpful responses from the teachers surveyed during the user study; these responses will help us design future iterations of this algorithm. While the user study was insightful and validated a lot of the work we've presented in this thesis, the true test of this framework will be in the form of a classroom study. Overall, the system is very promising; Retain uses a novel approach to solve major problems related to long-term retention in educational settings.

# Bibliography


[1] Diane F Halpern and Milton D Hakel. Applying the science of learning to the university and beyond: Teaching for long-term retention and transfer. *Change: The Magazine of Higher Learning*, 35(4):36–41, 2003.

[2] Andrew Sutherland. Quizlet. `https://quizlet.com/`.

[3] Frank N Dempster. The spacing effect: A case study in the failure to apply the results of psychological research. *American Psychologist*, 43(8):627, 1988.

[4] Arthur W Melton. The situation with respect to the spacing of repetitions and memory. *Journal of Verbal Learning and Verbal Behavior*, 9(5):596–606, 1970.

[5] Frank N Dempster. Time and the production of classroom learning: Discerning implications from basic research. *Educational Psychologist*, 22(1):1–21, 1987.

[6] Douglas L Hintzman. Theoretical implications of the spacing effect. 1974.

[7] Piotr A.Wozniak. Optimization of learning. Master's thesis, University of Technology in Poznan, 1990.

[8] SuperMemo World. Supermemo. `https://supermemo.com`.

[9] Supermemo: Creating a knowledge tree structure in the contents window.

[10] Damien Elmes. Anki. `https://apps.ankiweb.net/`.

[11] Peter Bienstman. Mnemosyne. `http://mnemosyne-proj.org/`.

[12] Jun Liu, Lu Jiang, Zhaohui Wu, Qinghua Zheng, and Yanan Qian. Mining learning-dependency between knowledge units from text. *The VLDB Journal*, 20(3):335–345, 2011.

[13] Chen Liang, Zhaohui Wu, Wenyi Huang, and C Lee Giles. Measuring prerequisite relations among concepts. In *EMNLP*, pages 1668–1674, 2015.

[14] Shuting Wang, Chen Liang, Zhaohui Wu, Kyle Williams, Bart Pursel, Benjamin Brautigam, Sherwyn Saul, Hannah Williams, Kyle Bowen, and CL Giles. Concept hierarchy extraction from textbooks. In *The ACM Symposium on Document Engineering*, 2015.

[15] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*, pages 693–700. McGraw-Hill Higher Education, 3rd edition, 2009.

[16] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962.

[17] Diane J Hu, Rob Hall, and Josh Attenberg. Style in the long tail: Discovering unique interests with latent variable models in large scale social e-commerce. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1640–1649. ACM, 2014.

# Appendix A

# Implementation of User Interface for Sample Example

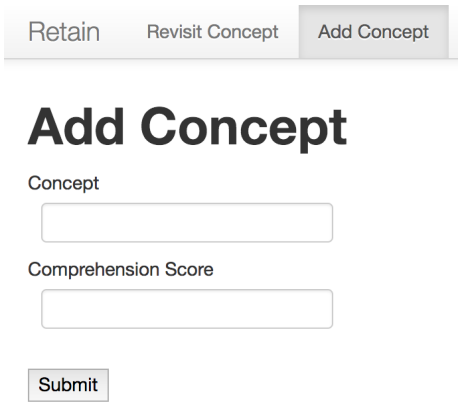



Figure A.1: UI for adding a new concept to the system.

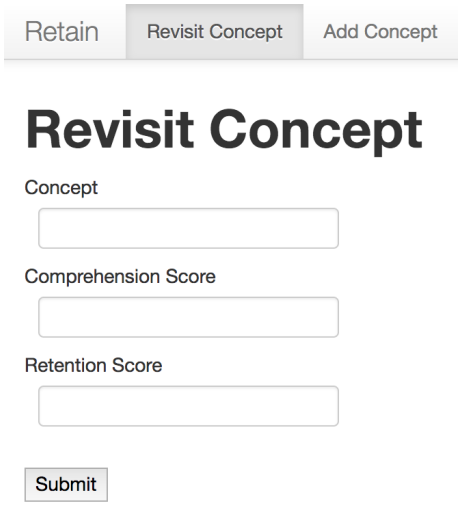



Figure A.2: UI for submitting comprehension and retention scores for a concept the system has prompted the student to revisit.

# Appendix B

# Qualitative Evaluation Questionnaire

Suppose you are tutoring a student for a Business Mathematics course.

The course has a concept dependency graph which gives you an idea of which concepts depend on other concepts. In these graphs, if there is an arrow from concept A to concept B, it means that before one can learn concept A, they need to have a relatively sufficient mastery of concept B.

Your teaching methodology is informed by spaced repetition. This means that you periodically revisit topics. Your revisiting schedule is determined by a spaced repetition algorithm. You also regularly administer quizzes to your student in order to gauge his comprehension on various topics. You continuously update your records of your students' comprehension of each topic using these quiz scores. In this framework, it's possible for a student to have a good handle of a concept, and then after several time units, no longer have a good handle on the concept.

We will use the following legend:

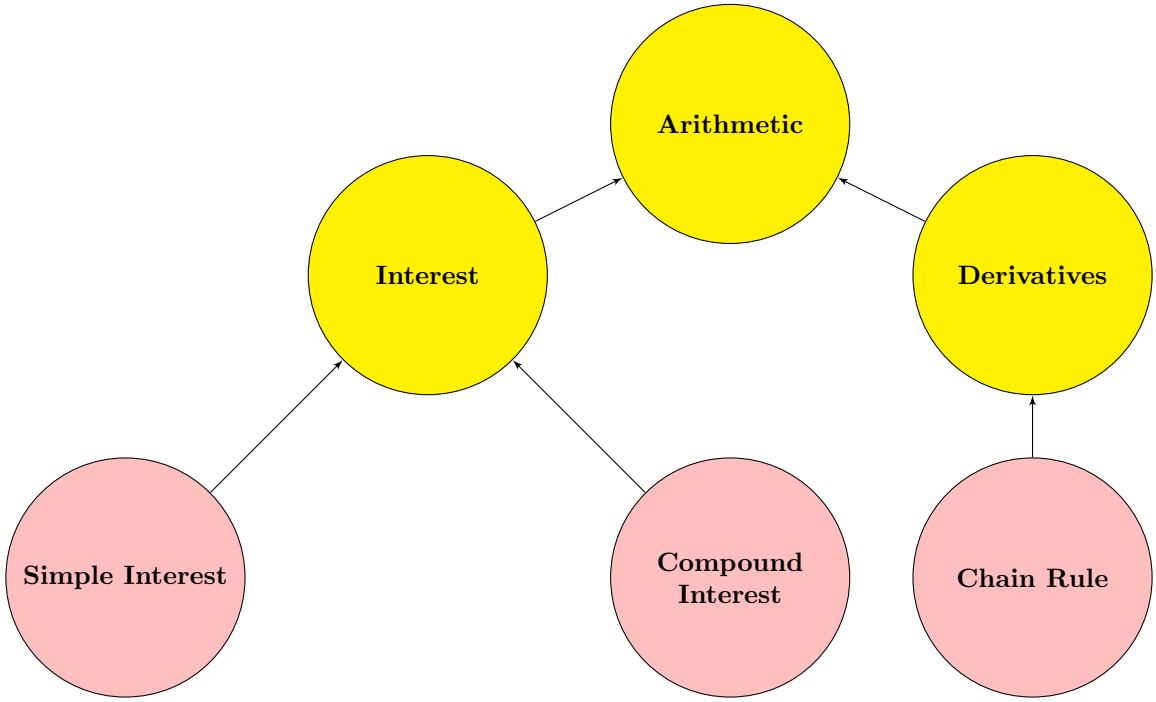
Most recent quiz score for concept was low

Spaced repetition algorithm says that it's time to review this concept

**Scenario One**

This particular concept graph encodes the following information: as far as your most recent records can indicate that your student seems to have forgotten some of the fundamentals of arithmetic, derivatives and what interest actually is; as such, their most recent quiz scores in those two areas have been low. The SM-2

spaced repetition algorithm recommends that you have your student revisit the following concepts: simple interest, compound interest, and the chain rule.




1. When choosing which concepts to recommend to the user, what types of things would you consider? What types of things would you like to consider when recommending concepts to a student under the spaced repetition paradigm?
2. Which concept do you think the student would gain the most from revisiting? Why?:
   (a) Simple Interest
   (b) Interest
   (c) Compound Interest
   (d) Arithmetic
   (e) Derivatives
   (f) Chain Rule
   (g) I don't know
3. Which of the following would you prefer to recommend? Why?
   (a) Interest
   (b) Arithmetic
   (c) I don't know
4. Which of the following concept recommendations makes the most sense to you? Why?

(a) Simple Interest, Compound Interest, Chain Rule

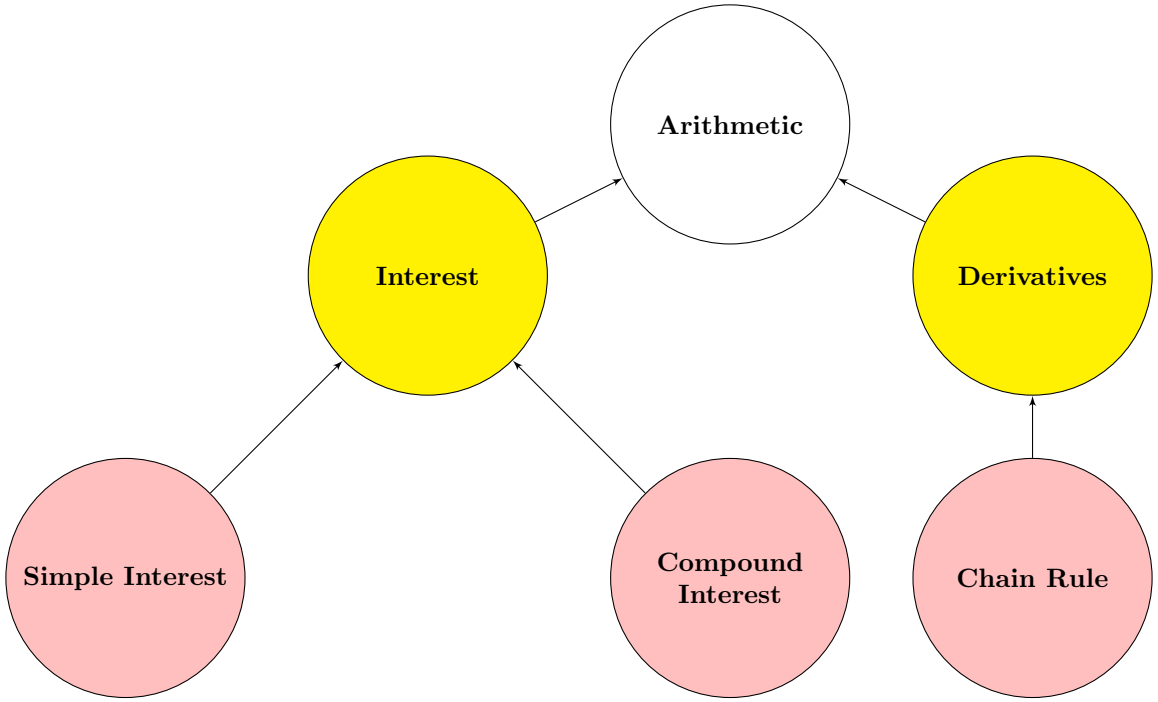
(b) Arithmetic, Interest, Derivatives

(c) Simple Interest, Derivatives, Compound Interest

(d) I don't know

**Scenario Two**

This particular concept graph encodes the following information: as far as your most recent records can indicate, your student has a good mastery of arithmetic. Unfortunately, over time, they seem to have forgotten some of the fundamentals of derivatives and what interest actually is; as such, their most recent quiz scores in those two areas have been low. The SM-2 spaced repetition algorithm recommends that you have your student revisit the following concepts: simple interest, compound interest, and the chain rule.




1. Which concept do you think the student would gain the most from revisiting? Why?:

   (a) Simple Interest

   (b) Interest

   (c) Compound Interest

   (d) Arithmetic

   (e) Chain Rule

   (f) Derivatives

   (g) I don't know

2. Which of the following would you prefer to recommend? Why?

   (a) Interest

   (b) Derivatives

   (c) I don't know

3. Which of the following concept recommendations makes the most sense to you? Why?

   (a) Simple Interest, Compound Interest, Chain Rule

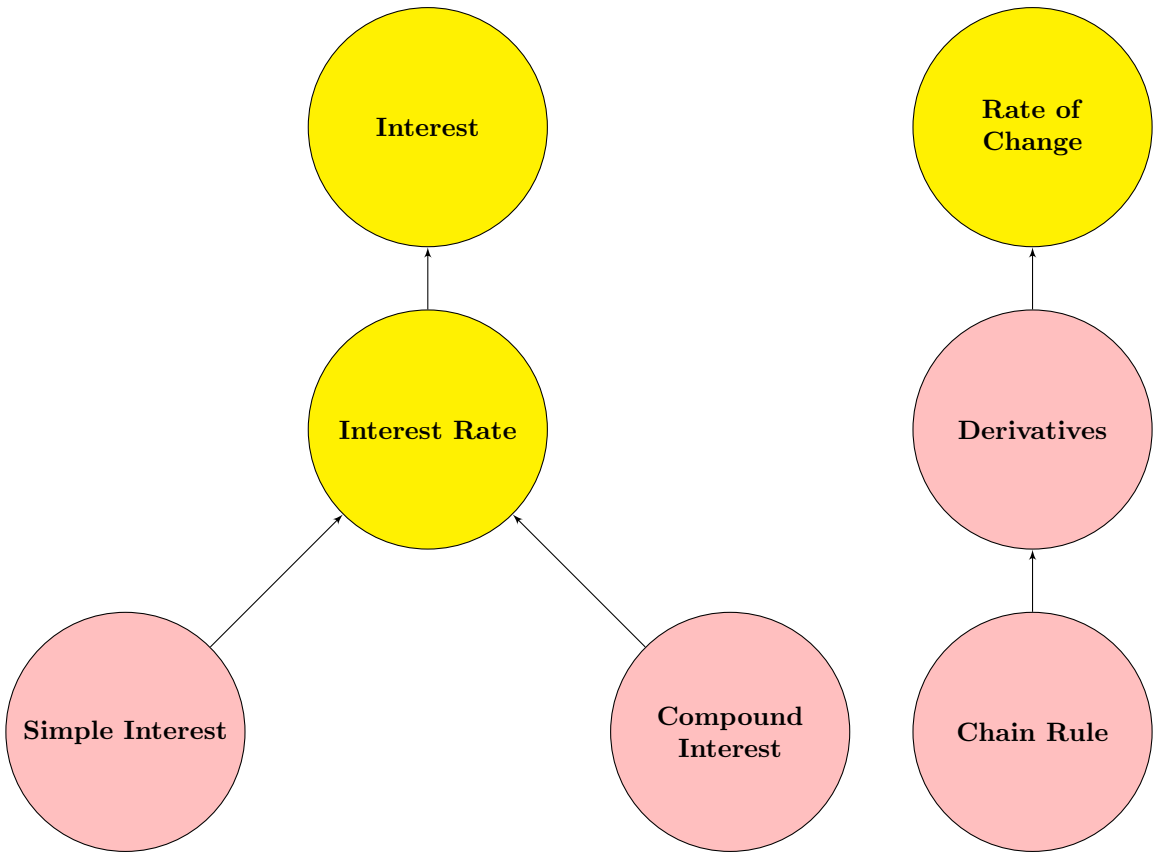
   (b) Interest, Derivatives, Simple Interest

   (c) Arithmetic, Simple Interest, Interest

   (d) I don't know

**Scenario Three**

This particular concept graph encodes the following information: it seems that they have forgotten some of the fundamentals of interest, interest rate, and rate of change; as such, their most recent quiz scores in those two areas have been low. The SM-2 spaced repetition algorithm reminds you that you should consider revisiting the following concepts with your student: simple interest, compound interest, derivatives, and the chain rule.



1. Which concept do you think the student would gain the most from revisiting? Why?:
    (a) Interest
    (b) Interest Rate
    (c) Simple Interest
    (d) Compound Interest
    (e) Rate of Change
    (f) Derivatives
    (g) Chain Rule
    (h) I don't know
2. Which of the following would you prefer to recommend? Why?
    (a) Interest
    (b) Rate of Change
    (c) I don't know
3. Which of the following concept recommendations makes the most sense to you? Why?
    (a) Simple Interest, Chain Rule, Rate of Change
    (b) Compound Interest, Chain Rule
    (c) Interest, Rate of Change, Interest Rate
    (d) I don't know

# Appendix C

# IRB Exemption Documentation

Below, is documentation certifying that an IRB exemption was obtained for the user study discussed in this paper.

**IRB EXEMPT APPROVAL**

**RPI Name: Dr. ChengXiang Zhai**

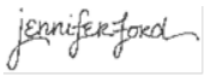
**Project Title: *RETAIN: Leveraging Spaced Repetition to Build a Concept Recommendation System in the Interest of Improving Retention in Educational Settings***
**IRB #: 17715**
**Approval Date: April 20, 2017**

Thank you for submitting the completed IRB application form and related materials. Your application was reviewed by the UIUC Office for the Protection of Research Subjects (OPRS). OPRS has determined that the research activities described in this application meet the criteria for exemption at 45CFR46.101(b)(1). This message serves to supply OPRS approval for your IRB application.

Please contact OPRS if you plan to modify your project (change procedures, populations, consent letters, etc.). Otherwise you may conduct the human subjects research as approved for a period of five years. Exempt protocols will be closed and archived at the time of expiration. Researchers will be required to contact our office if the study will continue beyond five years.

We appreciate your conscientious adherence to the requirements of human subjects research. If you have any questions about the IRB process, or if you need assistance at any time, please feel free to contact me at OPRS, or visit our website at http://oprs.research.illinois.edu

Sincerely,



Human Subjects Research Specialist, Office for the Protection of Research Subjects

Attachment(s): research team attachment, approved consent document, approved waiver of documentation of informed consent

C: Shilpa Subrahmanyam

Office of the Vice Chancellor for Research | Office for the Protection of Research Subjects
University of Illinois | Urbana-Champaign

805 West Pennsylvania Avenue, MC-095 | Urbana, IL 61801